InHolland

VHDL

S88-n Protocol in VHDL

Authors:

Koen Groot
549543

Ruben Pera
551198

Samenvatting

In dit onderzoeks verslag wordt de werking van het s88-n protocol onderzocht. Deze werking wordt geïmplementeerd met behulp van VHDL.

Inhoudsopgave

1 Inleiding **4**

2 Probleem Omschrijving **5**
2.1 Probleem Omschrijving 5
2.2 Hoofdvraag 5
2.3 Deelvragen 6

3 Specificaties **7**
3.1 Hardware 7
3.2 Software 7

4 Vereisten **8**

5 Analyse **9**
5.1 deelvragen 9
5.1.1 Hoe werkt het s88 protocol? 9
5.1.2 Hoe wordt het XILINX Spartan 3E FPGA bord gebruikt? 11
5.1.3 Hoe kan de hardware beschrijving taal VHDL gebruikt worden op het XILINX Spartan 3E FPGA bord? 12

6 Ontwerp **13**
6.1 Textueel ontwerp van het programma 13
6.2 Custom Clockrate genereren 14
6.3 De S88 Signalen 15
6.4 Initialisatie van het schuifregister 15
6.5 Uitlezen van het Schuifregister 18

7 Implementatie **21**
7.1 VHDL uitgelegd 21
7.2 Declaratie VHDL 22
7.3 Initialisatie van het schuifregister 22
7.4 Uitlezen van het Schuifregister 23

8 Apendix **24**

Hoofdstuk 1

Inleiding

test2

Hoofdstuk 2

Probleem Omschrijving

2.1 Probleem Omschrijving

De RM88 is ontworpen om een goedkope feedback bus te zijn, de feedback wordt geleverd via het S88 protocol.
Het idee is eenvoudig, de RM88 is een serieel schuif register met een parralelle load input, de input zal dus binnenkomen via het timingsprotocol S88.
In dit onderzoek gaat er onderzocht worden of dit nagebouwd kan worden met een XILINX Spartan 3E FPGA bord en de harware omschrijf taal VHDL.

2.2 Hoofdvraag

De Hoofdvraag bij dit onderzoek luidt:

Hoe kan er op een XILINX Spartan 3E FPGA bord met behulp van de hardware beschrijving taal VHDL een s88 protocol geïmplementeerd worden?

2.3 Deelvragen

Om de Hoofdvraag goed te kunnen beantwoorden zijn er de volgende deelvragen opgesteld:

1. Hoe werkt het s88 protocol?
2. Hoe werkt een XILINX Spartan 3E FPGA bord?
3. Hoe kan de hardware beschrijving taal VHDL gebruikt worden op het XILINX bord?
4. Hoe kan een schuifregister aangesloten worden op het XILINX bord?

Pas nadat alle deelvragen beantwoordt zijn kan op de hoofdvraag een volledig antwoord geformuleerd worden.

Hoofdstuk 3

Specificaties

In de specificaties wordt omschreven waar het systeem aan moet voldoen bij het afleveren van het systeem.

3.1 Hardware

Er wordt gebruik gemaakt van de volgende Hardware

- Er wordt gebruik gemaakt van een XILINX Spartan 3E FPGA bord.
- Er wordt gebruik gemaakt van een Field Programmable Gate Array (FPGA).

3.2 Software

Er wordt gebruik gemaakt van de volgende Software

- Er wordt gebruik gemaakt van een 64 bit versie van Windows 7.
- Er wordt gebruik gemaakt van Oracle VM VirtualBox om de Windows 7 in te emuleren.
- Er wordt gebruik gemaakt van XILINX Tools voor Windows.

Hoofdstuk 4

Vereisten

Om te kunnen onderzoeken of

De volgende vereisten zijn gesteld aan het te opleveren systeem.

- Het XILINX Spartan 3E FPGA bord moet kunnen communiceren met een Field Programmable Gate Array (FPGA).
- De communicatie tussen het XILINX bord en de FPGA moet plaatsvinden door middel van het S88 protocol.
- De communicatie tussen het XILINX bord en de FPGA wordt als volgt beschreven, het XILINX bord zal via het S88 protocol de eerste byte uit het FPGA lezen.
- Het XILINX bord zal geconfigureerd worden met de hardware beschrijving taal VHDL.

Hoofdstuk 5

Analyse

In dit onderzoek gaat er onderzocht worden of het S88 protocol nagebouwd kan worden met een XILINX Spartan 3E FPGA bord en de harware omschrijf taal VHDL.

5.1 deelvragen

Voor de Analyse zijn er meerdere deelvragen opgesteld,

1. Hoe werkt het s88 protocol?
2. Hoe werkt een XILINX Spartan 3E FPGA bord?
3. Hoe kan de hardware beschrijving taal VHDL gebruikt worden op het XILINX bord?
4. Hoe kan een schuifregister aangesloten worden op het XILINX bord?

5.1.1 Hoe werkt het s88 protocol?

Het s88 wordt uitgelegd met behulp van onderstaande afbeelding.

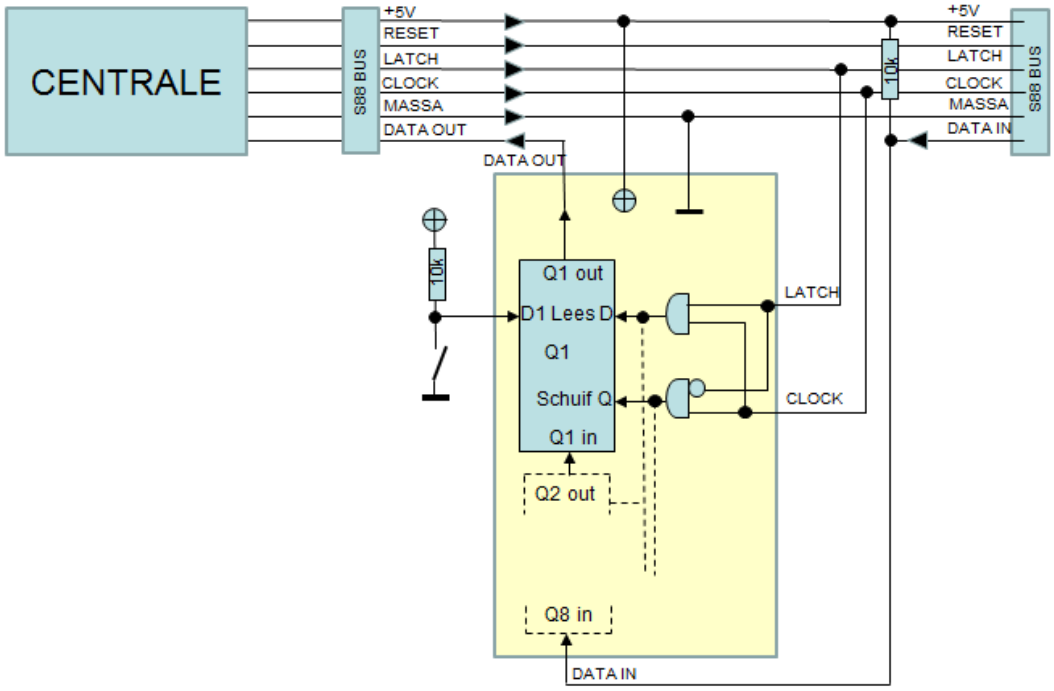
Hierin is CENTRALE het in dit project gebruikte XILINX bord.

Een centrale, die hier CENTRALE heet, kan met behulp van het s88 protocol de rechter bus uitlezen. Dit gebeurt met behulp van een schuifregister.

Met de DATA OUT poort van de CENTRALE BUS wordt data ingelezen. Deze is verbonden met de Q1 out poort van het schuifregister, hiermee wordt dus informatie van het schuifregister naar de CENTRALE overgedragen.

De werking is als volgt, op het moment dat de CENTRALE een positieve flank geeft op de CLOCK en de LATCH hoog is, dan zal het schuifregister de data op Ingang D inlezen. Geeft de CENTRALE een positieve flank en is de LATCH Laag, dan zal het schuifregister één positie opschuiven.

Door een puls op RESET te geven worden de buffers gereset en zijn deze weer gereed voor het ontvangen van nieuwe data.



@onlineID,
title = S88 TERUGMELDERS,
date = 04-03-2016,
url = http://users.telenet.be/RedDeBist/MBAAN/S88%20terugmelder.htm

5.1.2 Hoe wordt het XILINX Spartan 3E FPGA bord gebruikt?

5.1.3 Hoe kan de hardware beschrijving taal VHDL gebruikt worden op het XILINX Spartan 3E FPGA bord?

Hoofdstuk 6

Ontwerp

Nu alle deel vragen beantwoord zijn in de Analyse kan er verder gegaan worden met het ontwerpen van de werking van het XILINX bord.

6.1 Textueel ontwerp van het programma

De volgende elementen moeten ontworpen en gerealiseerd worden:

1. De clock naar beneden schalen zodat de Leds door mensen te zien zijn
2. De S88 Signalen moeten nagebootst worden

In dit hoofdstuk zal het ontwerpen plaatsvinden, dus in woorden uitgelegd wat er gedaan moet worden, en in het hoofdstuk realiseren wordt het in VHDL gerealiseerd.

6.2 Custom Clockrate genereren

Het eerste wat gedaan moet worden is de clockrate naar beneden schalen, dit wordt gedaan omdat de normale clockrate 50 MHz is.

Deze clockrate is te hoog om met een menselijk oog te kunnen waarnemen, hierom zal de clockrate intern verlaagt moeten worden.

Dit zal gebeuren door middel van een interne timer, die iedere clocktick met één verhoogd wordt.

Op het moment dat deze timer, die vanaf dit moment ClockTimer genoemt zal worden, een waarde van 2.5×10^4 bereikt zal er een andere variabele genaamt CustomClock met getoggled worden.

Dit houdt in dat er een eigen clock wordt gegenereerd die 0.5 seconde hoog is om daarna 0.5 seconde laag te zijn. Dit betekend dat de gegeneerde clock ofwel CustomClock een clockrate van 1 Hz heeft.
Dit is goed zichtbaar voor het menselijke oog.

6.3 De S88 Signalen

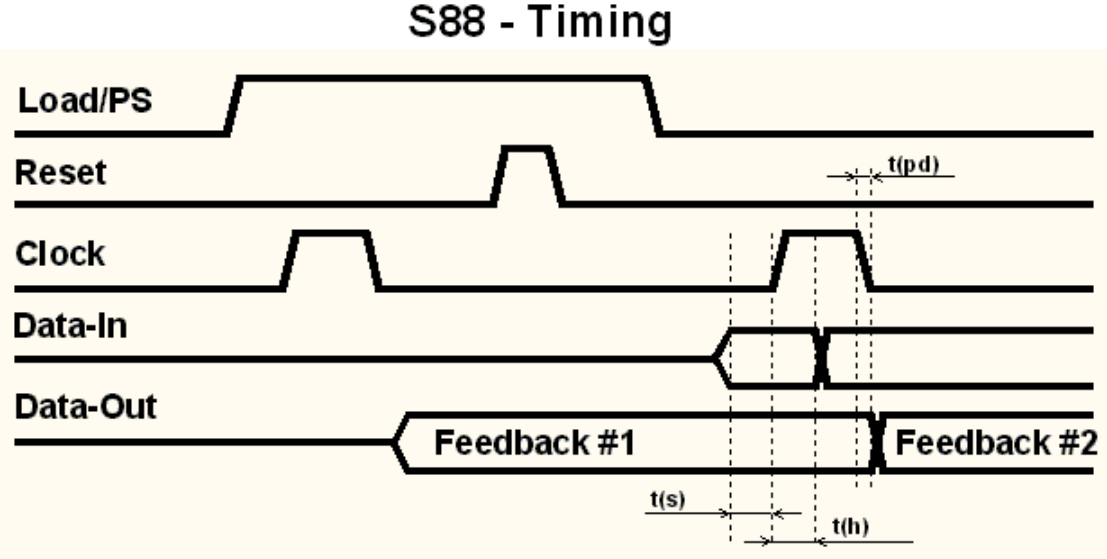
In de Analyze in de sectie ”Hoe werkt een s88 protocolïs reeds uitgelegd hoe het s88 protocolr werkt.
Hier is uit af te leiden dat er door er gebruikt gemaakt wordt van de volgende IO poorten:

- CLOCK
- DATA-OUT
- LOAD
- RESET

Het ontwerpen van deze signalen wordt opgesplitst in twee secties, De initialisatie en het daadwerkelijke uitlezen van het schuifregister.

6.4 Initialisatie van het schuifregister

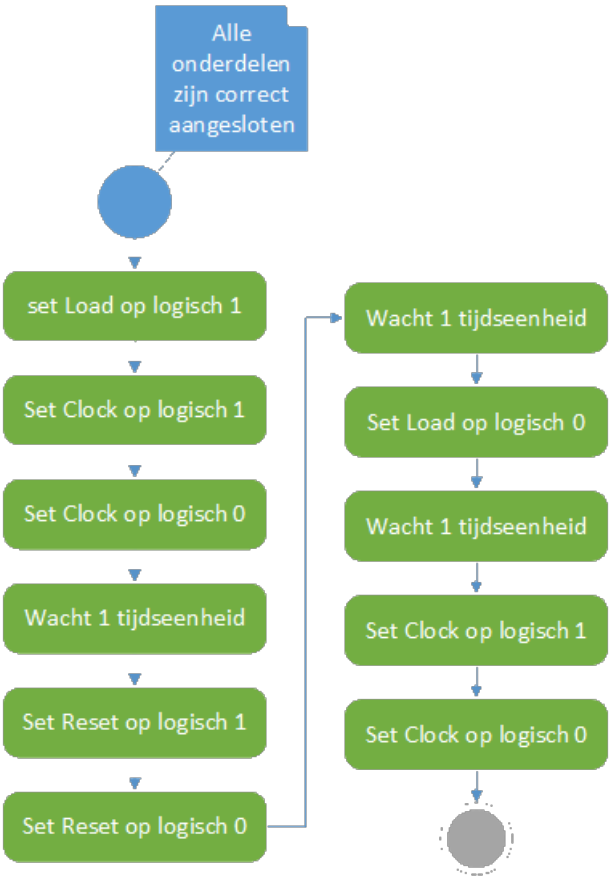
De volgorde van deze signales wordt beschreven in onderstaande image.



Deze image geeft aan welke poorten wanneer op een logisch 1 gezet moeten worden. Als voorbeeld, de lijn onder Load/PS is de lijn die bij

Load hoort. In het begin geeft de lijn een logisch 0 weer, de stijding betekent dat daar de poort op een logisch 1 gezet wordt.

Van de poorten Load/PS, Reset en Clock is een UML activiteiten diagram gemaakt, dit is gedaan zodat de volgorde van de signalen overzichtelijker weer te geven is.



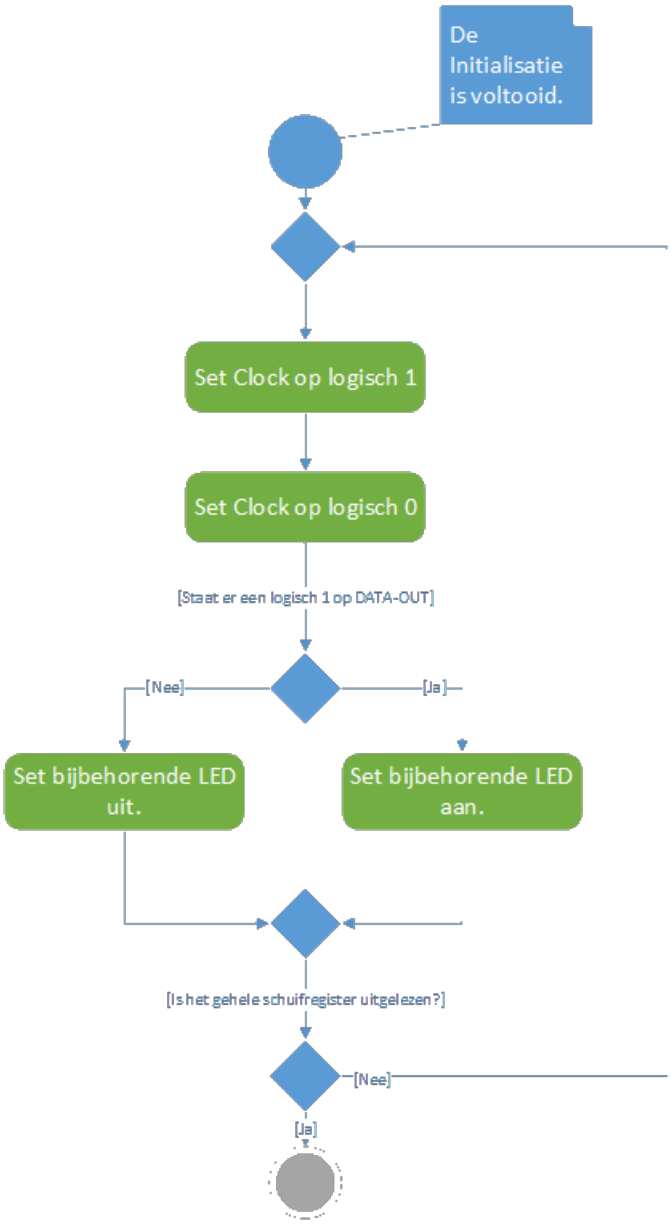
Hierin staat één tijdseenheid voor één seconde en is de duur van elke opdracht gelijk aan één tijdseenheid, dus één seconde.
Het moet gelezen worden door middel van de pijlen waarin er bovenaan begonnen met lezen wordt bij de blauwe cirkel.
Er mag alleen begonnen aan de opdracht als "Alle onderdelen zijn correct aangesloten"waar is, anders heeft het uitvoeren van de in de diagram beschreven opdrachten geen zin.

Als er dus gecontroleerd is dat alles correct aangesloten is kan er begonnen worden met de eerste opdracht, in dit geval ”Set Load op logisch 1”. Nadat deze opdracht uitgevoerd is kan er verdergegaan worden met de opdracht die door middel van de pijl verbonden is, in dit geval ”Set Clock op logisch 1”. Zo wordt er verdergegaan totdat alle opdracht uitgevoerd zijn, als alle opdrachten uitgevoerd zijn is het schuifregister succesvol geïnitialiseerd.

6.5 Uitlezen van het Schuifregister

Nadat de initialisatie voltooid is kan er begonnen worden met het uitlezen van het schuifregister. Zoals is beschreven in de analyse kan het schuifregister op de volgende manier uitgelezen worden, er wordt één tijdseenheid een logische 1 op Clock gezet, op de falling-edge van de Clock komt de volgende bit van het schuifregsiter op Data-Out gezet.

Dus nadat de Data-Out uitgelezen te hebben kan door middel van de Clock één tijdseenheid een logisch 1 te maken het volgende bit uit het schuifregister op de Data-Out gezet worden. En op die manier kan het hele schuifregister uitgelezen worden.



Wat hier niet goed weergegeven is is dat het net zo vaak uitgevoert zal worden totdat alle bits uit het schuifregister uitgelezen zijn, normaal gesproken wordt dit gedaan met een Loop maar omdat er bij de vereisten al vastgelegd is dat er gebruik gemaakt zal worden van de hardware beschrijving taal VHDL zal dit anders verlopen.

VHDL beschikt namelijk wel degelijk over een loop maar deze zal hier niet gebruikt worden, de redenatie hierachter is als volgt. Alle elementen worden geclockt op de CustomClock. Een loop kan niet geclockt worden en zal dus zal niet goed kunnen omgaan met de rest van het systeem.

Wat er wel gedaan zal worden is het volgende, vanaf het moment dat de initialisatie voltooid is zal er om de tijdseenheid de Clock voor één tijdseenheid logisch 1 gemaakt worden. De tijdseenheid die er dan tussen zit zal gebruikt worden om Data-Out uit te lezen. Dit zal net zo lang doorgaan totdat alle bits uit het schuifregister door het systeem uitgelezen zijn.

Hoofdstuk 7

Implementatie

Voordat er begonnen kan worden wordt er eerst een korte uitleg gegeven over de structuur, syntax en code van de hardware beschrijving taal VHDL.

7.1 VHDL uitgelegd

VHDL begint met een declaratie van alle gebruikte libraries, dit zijn bibliotheken waarin staat hoe de VHDL code met het XILINX bord om moet gaan, bijvoorbeeld waar de aansluitingen zitten.

Na het declareren van de libraries moeten alle IO poorten van het systeem gedeclareerd worden.

Na het declareren van de libraries en de IO poorten kan er nog gebruikt gemaakt worden van interne variabelen, signals genoemt. Waar de IO poorten alleen binair 0 of 1 kunnen zijn kunnen signals meer informatie bevatten, het is mogelijk om integers, bools en dergelijke te gebruiken.

Nadat alles gedeclareerd is kan er begonnen worden met de daadwerkelijke functies.

7.2 Declaratie VHDL

De volgende libraries moeten gedeclareerd worden om de VHDL te kunnen communiceren met het XILINX bord.

- *ieee*;
- $ieee.std_logic_1164.all$;
- $IEEE.std_logic_unsigned.all$;
- $ieee.numeric_std.all$;

7.3 Initialisatie van het schuifregister

Hieronder wordt een lijst gegeven van alle ***uitgangen*** die gebruikt gaan worden:

- LED0 t/m LED7.
  Deze LEDs worden gebruikt om de toestand van de uit Data-Out ingelezen bits te weer te geven, ofwel 1(Led aan) ofwel 0(led uit).
- GPIO16
  Dit is de IO poort waarop Load aangesloten wordt.
- GPIO17
  Dit is de IO poort waarop Reset aangesloten wordt.
- GPIO14
  Dit is de IO poort waarop Clock aangesloten wordt.

Hieronder wordt een lijst gegeven van alle ***Ingangen*** die gebruikt gaan worden:

- GPIO12
  Dit is de IO poort waarop Data-Out aangesloten wordt.
- SW0
  Dit is de IO poort die aangeeft of de upper of lower byte gelezen wordt.
- Onboardclock
  Dit is de eerdergenoemde interne clock, deze is geklokt op 50 MHz.

7.4 Uitlezen van het Schuifregister

Hoofdstuk 8

Apendix

```
1  Library ieee;
2  use ieee.std_logic_1164.all;
3  use IEEE.std_logic_unsigned.all;
4  use ieee.numeric_std.all;
5
6  --library UNISIM;
7  --use UNISIM.VComponents.all;
8  entity s88V2 is
9      port
10     (
11         OnboardClock, SW0, GPIO12 : in std_logic;
12 --       LED0, LED1, LED2, LED3, LED4, LED5, LED6,
       LED7, GPIO14, GPIO16, GPIO17 : out std_logic;
13
14
15         clock           : in std_logic;
16         data_input      : in std_logic;
17         load_output     : out std_logic := '0';
18         clock_output    : out std_logic := '0';
19         reset_output    : out std_logic := '0';
20         data_output     : out std_logic := '0';
21         test_output     : out std_logic := '0'
22     );
23 end s88V2;
```

```
24
25
26 architecture s88TimingV2 of s88V2 is
27 signal CounterForTheClock   : integer := 0;
          -- gebruikt om de hoeveelheid klokslagen bij
      te houden
28 signal CounterForTheTime    : integer := 0;
          --
29 signal HertzToUse           : integer := 1;
30 begin
31
32     increase_CounterForTheClock_on_clock : process(
          clock)
33     begin
34         if rising_edge(clock) then
35             if CounterForTheClock = HertzToUse then
36                     --als er de gewenste hoeveelheid
                           klokslagen verstreken zijn
                        dan wordt de
                        CounterForTheTime verhoogd.
37                 if CounterForTheTime > 8 then
38                     CounterForTheTime <= 0;
39                 else
40                     CounterForTheTime <=
                           CounterForTheTime + 1;
41                 end if;
42                 CounterForTheClock <= 0;
43             else
44             CounterForTheClock <= CounterForTheClock
                   + 1; --houdt bij hoeveel klokslagen
                  er zijn geweest,
45             end if;
46         end if;
47
48     end process increase_CounterForTheClock_on_clock
          ;
49
50     Read_IO_Case : process(CounterForTheTime)
```

```
51          begin
52              case CounterForTheTime is
53                  when 0 =>
54                      --doe niks
55                  when 1 =>
56                      load_output     <= '1';
57                  when 2 =>
58                      clock_output    <= '1';
59                  when 3 =>
60                      clock_output    <= '0';
61                      data_output <= '1';
62                  when 4 =>
63                      reset_output    <= '1';
64                  when 5 =>
65                      reset_output    <= '0';
66                  when 6 =>
67                      load_output     <= '0';
68                  when 7 =>
69                      clock_output    <= '1';
70                  when 8 =>
71                      clock_output    <= '0';
72                      data_output <= '0';
73                  when others =>
74                      --doe niks
75              end case;
76          end process Read_IO_Case;
77
78  end s88TimingV2;
```